TÓM TẮT CÁC KIẾN THỨC

CẦN NHỚ TRONG LẬP TRÌNH

Tài liệu có tham khảo:

Sách: Nhập môn lập trình (Thầy Trần Đan Thư,…)

Sách: Kĩ thuật lập trình (Thầy Trần Đan Thư,…)

Slide bài giảng của Trường Đại học Công nghệ Thông tin – ĐHQG TP HCM

Slide bài giảng của Thầy Đặng Bình Phương - Trường Đại học Khoa học Tự nhiên – ĐHQG TP HCM

Và các nguồn tài liệu trên mạng…

https://thanhcuong.wordpress.com/2011/12/10/nh%E1%BB%AFng-khc-bi%E1%BB%87t-c-th%E1%BB%83-b%E1%BA%A1n-ch%C6%B0a-bi%E1%BA%BFt-trong-l%E1%BA%ADp-trnh-c-c/

tìm hiểu ngôn ngữ scala

1. **MỘT SỐ KHÁI NIỆM & TÍNH CHẤT:**

* **Lập trình**: nghệ thuật cài đặt một hay nhiều thuật toán trừu tượng có liên quan với nhau bằng một ngôn ngữ lập trình để tạo ra một chương trình máy tính.
* **Thuật toán**: là tập hợp (dãy) hữu hạn các chỉ thị (hành động) được định nghĩa rõ ràng nhằm giải quyết một bài toán cụ thể nào đó.
* **5 tính chất**:
* Tính chính xác: *quá trình tính toán hay các thao tác máy tính thực hiện là chính xác.*
* Tính rõ ràng: *các câu lệnh minh bạch được sắp xếp theo thứ tự nhất định.*
* Tính khách quan: *được viết bởi nhiều người trên máy tính nhưng kết quả phải như nhau.*
* Tính phổ dụng: *có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.*
* Tính kết thúc: *hữu hạn các bước tính toán.*
* **Biểu diễn thuật giải bằng:**
* *Ngôn ngữ tự nhiên*
* *Sơ đồ - lưu đồ khối*
* *Mã giả*

1. **CÁC KIỂU DỮ LIỆU CƠ BẢN:**

***Trong C/C++: (chạy trên Visual Studio 2015)***

Có 4 kiểu dữ liệu cơ bản (*Primitive Types*):

* *Kiểu luận lí*: boolean – ***bool.*** *Để xuất ra từ* ***“true”*** *hoặc* ***“false”*** *mà không phải là các số* ***0*** *và* ***1****, thêm* ***boolalpha*** *trước khi xuất, ngược lại thêm* ***noboolalpha****. Giá trị* ***khác 0*** *mặc định là* ***1*** *(****true****).*
* *Kiểu ký tự*: ***char*** (*cũng là kiểu số nguyên?*) – lưu ký tự *1byte dựa theo bảng mã ASCII* và ***wchar\_t*** – lưu ký tự *2byte dựa trên bảng mã quốc tế UTF-16* (1 dạng mã Unicode).
* *Kiểu số nguyên*: ***int*** – với nhiều cỡ cho số có dấu (*signed*) và không dấu (*unsigned*); *kiểu nguyên 64-bit* **\_*int64*** *(C/C++ không hỗ trợ các định dạng nhập xuất có sẵn, phải nhập gián tiếp thông qua nhập xuất kiểu double, rồi sau đó ép kiểu lại)*
* *Kiểu số thực*: ***float, double***

Ngoài ra còn các kiểu dữ liệu khác như: ***enum*** (kiểu thứ tự), ***record*** (bản ghi), ***union*** (kiểu đơn vị), ***pointer*** (con trỏ), ***array*** (mảng) và các cấu trúc dữ liệu khác do người dùng định nghĩa (***struct, linked list, double linked list, stack, queue, tree***, …).

***Lưu ý:***

-Độ lớn của các kiểu *phụ thuộc* vào ***IDE*** (*Integrated Development Environment* *– Môi trường phát triển chương trình tích hợp*) => dùng lệnh sizeof() để kiểm tra.

-Ngoài các ***phép toán số học*** còn có các ***phép toán trên bit*** (**&**, **|**, **^**, **~**, **<<**, **>>**) và chúng cho *kết quả khác nhau*.

-Trong các chương trình ứng dụng quan trọng nên cẩn thận khi so sánh các số thực: biểu thức a != b luôn được thay thế bởi fabs(a - b) < epsilon với epsilon là một số khá nhỏ như một phần vạn hay một phần triệu.

* **Ép kiểu: 2 loại**
* **Không tường minh (***ngầm định***):** compiler ***tự động*** chuyển kiểu phù hợp, cụ thể là các biến có giá trị nhỏ hơn được chuyển về kiểu có độ ưu tiên/ giá trị cao nhất trong biểu thức;
* **Tường minh:** người lập trình chủ động chuyển, có **5** loại
* **C-style casts:**

vd: ***int inum*** = 5; cout << **(float)*inum***; *hoặc* cout << **float(*inum*);**

* *Không kiểm tra tính đúng sai của biểu thức trong quá trình compile.*
* **Static casts (C++):**

Vd: ***char c***= ‘A’; cout << ***static\_cast*<***int***>**(***c***); //in ra số 65-mã ASCII của A

* *Kiểm tra tính đúng sai của biểu thức trong quá trình compile.*
* **Const casts**
* **Dynamic casts**
* **Reinterpret casts**

1. **THƯ VIỆN, HÀM CÓ SẴN & MỘT SỐ CÂU LỆNH CƠ BẢN:**

* **#include <……>** khai báo thư viện

*C: stdio.h, conio.h, stdlib.h, math.h, ctype.h…*

*C++: iostream, cmath, string, fstream, ctime…*

* **Hàm toán học** (*cmath - C++ chuẩn, math.h - C/C++ không chuẩn*): ***pow(x), exp(x), floor(x), ceil(x), fmod*** (*thay cho toán tử* ***%***)…
* **Hàm ký tự**: *#include <ctype.h>*, ***bool*** ***isupper(char ch)***/ ***islower(char ch)***, ***char toupper(char)***/ ***tolower(char)***: kiểm tra ký tự ch có là ký tự hoa/thường hay không, trả về ký tự hoa thường. (*iswupper, iswlower*,…)
* **Lệnh/ từ khóa**: ***scanf/ scanf\_s/ cin/ gets*** (*nhập*), ***printf/ cout/ puts*** (*xuất*), ***getch()/\_getch()/ system(“pause”)*** (*chờ nhập 1 phím từ bàn phím/ dừng màn hình*)

**Câu lệnh lặp:**

* + - 1. ***for (… ;… ;…) {}***
      2. ***while(…){}***

***3. do***

***{}while(…);***

***4. goto***

**Câu lệnh rẽ nhánh:**

1. ***if(…) {}***

***else {}***

1. ***switch(…)***

***case:***

***Lưu ý:***

* Lệnh ***for*** có thể *bị khuyết*: vd for( ; … ; …), for( ; ; ),…
* Lệnh ***while*** có thể *không được thực hiện lần nào* vì điều kiện sai từ ban đầu
* Lệnh ***do…while*** *thực hiện ít nhất 1 lần*
* Lệnh ***if ... else …*** có thể *thiếu* ***else***
* Lệnh ***switch() … case: …*** có thể có nhiều ***case*** nhưng *mỗi* ***case*** *phải khác nhau*, cần lưu ý và tận dụng lệnh ***break*** trong mỗi ***case*** vì ***switch*** sẽ nhảy đến ***case*** tương ứng và thực hiện đến khi nào gặp ***break*** hoặc cuối ***switch*** mới kết thúc, có thể thiếu ***default***.
* Đều là lệnh đơn, có thể lồng nhau.

1. **HÀM:**

* Một chương trình có thể có nhiều hàm, nhưng chỉ có duy nhất một hàm main (***void main()*** *hoặc* ***int main()*** ).
* Kiểu trả về của *hàm con*: bất kì kiểu nào: int, char, float, void (*không cần return*)
* Tên hàm: theo quy tắc đặt tên định danh.
* Danh sách tham số: tham số hình thức đầu vào, giống khai báo biến, cách nhau bằng dấu phẩy, có thể truyền ở dạng *tham trị* hoặc *tham chiếu* tùy nhu cầu.
* Tầm vực: là *phạm vi hiệu quả của biến và hàm*.
* ***Biến toàn cục***: khai báo *ngoài tất cả các hàm*, *có tác dụng lên toàn chương trình*.
* ***Biến cục bộ***: khai báo *trong hàm hoặc khối* **{}** và *chỉ có tác dụng trong hàm hoặc khối đó (kể cả khối con của nó)*, sẽ *bị xóa khỏi bộ nhớ khi kết thúc khối khai báo nó*.
* **Truyền đối số** cho hàm:
* ***Truyền giá trị*** (*Call by Value*): dạng giá trị, có thể truyền *hằng, biến, biểu thức* nhưng hàm chỉ nhận giá trị, sử dụng khi *không có nhu cầu thay đổi giá trị của tham số* sau khi thực hiện hàm.
* ***Truyền tham chiếu*** (*Call by Reference – C++*): dạng địa chỉ, bắt đầu với dấu & trong khai báo, không truyền giá trị cho tham số này, sử dụng khi *có nhu cầu thay đổi giá trị của tham số* sau khi thực hiện hàm.
* ***Truyền địa chỉ*** (*Call by Adress*): dạng địa chỉ (con trỏ), không truyền giá trị cho tham số này, sử dụng khi *có nhu cầu thay đổi giá trị của tham số* sau khi thực hiện hàm.
* **Sử dụng tham chiếu là cách trả về giá trị cho chương trình.**
* **Cần cẩn thận khi làm việc.**
* **Nguyên mẫu hàm** (*prototype*)
* **Phân biệt** hàm và thủ tục: ***hàm*** có ***trả về giá trị*** (có *return*), ***thủ tục*** thì ***không***.
* **Định dạng dữ liệu nhập xuất với C:**
* **\\** (*xuất* **\**), **%%** (*xuất* **%**), **\r**, **\rn**, **\nr** (*xuống* *dòng*, tùy IDE), **\a** (tiếng chuông);
* **\e, \f, \v**; **%x** hay **%X** (*nhập xuất số nguyên dạng thập lục phân, xuất ra chữ thường/hoa*), **%o** (*nhập xuất số nguyên dạng bát phân*);
* **e** hay **E** (*nhập xuất số thực dạng viết số mũ dấu chấm động thay cho cơ số* **10**, vd *10.02***E***1998* là *10.02 x* ***10****1998*);
* nhập số double dùng **%lf, %le, %lE** => thay **l** thành **L** cho kiểu long;
* nhập số nguyên short (16 bit) dung **%hd, %hu, %hX**,… số nguyên dài: **%ld, %li, %lx**, …

***Quy định độ rộng và độ chính xác số thực:* %wid.pre**

vd **%3.5f** *(lấy 5 số sau dấu chấm)*

* **Định dạng dữ liệu nhập xuất với C++:**
* **#include <iostream** : nhập xuất chuẩn, …
* **#include <iomanip>**:

***setw(n)*** định độ rộng dữ liệu khi xuất, ***left, right*** dùng chung với ***setw(n)*** để canh lề

***setfill(ch)*** quy định ký tự ***ch*** thêm vào thay vì dùng *khoảng trống mặc định*

***hex*** (16 phân), ***dec*** (10 phân), ***oct*** (8 phân)

***Quy định độ rộng và độ chính xác số thực:* setprecision(n)**

với *n là tổng các chữ số*, thêm vào trước khi xuất.

* **Lưu ý:** trường hợp **biến tĩnh – biến cục bộ tĩnh**: (từ khóa static) được C/C++ *cấp phát vùng nhớ cố định trong suốt thời gian chạy* của chương trình, khi *hàm kết thúc* thì *biến tĩnh* được khai báo trong nó vẫn *chưa bị hủy*, vẫn *có thể được dùng trong những lời gọi hàm tiếp theo*.
* **Điểm khác nhau của /n và endln:**
* /n: không xóa bộ nhớ đệm. Dùng khi thao tác với file để nâng cao hiệu quả đọc ghi hoặc khi muốn giảm flush vùng nhớ đệm mỗi khi chạy.
* endln: xóa đi bộ nhớ đệm sau khi được gọi. Sử dụng khi muốn output có ngay lập tức và hạn chế thao tác với **file**.
* ***cout<< “VMT\n” << flush;*** *tương đương* ***cout<< “VMT”<<endln;***
* **Vấn đề trùng tên hàm** (*chồng hàm – function overloading* hay *đa xạ hàm – function polymorphism*): **phân biệt** nhờ kiểu tham số và số lượng các tham số
* **Hàm với giá trị mặc định của tham số**
* **Hàm có tham số là hàm**
* **Hàm có tham số kiểu dữ liệu**
* **Tập tin mã nguồn:**
* *\*.h*: tập tin khai báo
* *\*.c* hoặc *\*.cpp*: tập tin cài đặt mã
* Tập tin mã trung gian: *\*.obj* hay *\*.o*
* Dùng chỉ thị: **#ifndef … #endif** (chuẩn) hoặc **#pragma** (thường là *#pragma once* – không chuẩn) để *tránh việc trùng lặp các hằng số hoặc include file*,…
* Với nhiều tập tin mã nguồn, để một **biến** hay **hàm** được phép ***sử dụng chung*** thì thêm từ khóa extern, ngược lại thêm từ khóa static.

**Một số bài làm chơi** *(có đáp án tham khảo)***:**

[Hàm làm tròn số thực theo quy ước toán học (C/C++ có hỗ trợ hàm round() nhưng chỉ làm tròn lên phần nguyên)](http://codepad.org/6CIFaHmS)

[Hàm swap dùng template (cho mọi kiểu dữ liệu?)](http://codepad.org/CSGq0A4W)

[Bài: Vẽ tam giác Pascal](http://codepad.org/rXB8dVei)

[Dùng hàm đo thời gian chạy chương trình.](http://codepad.org/9pfP2dkl)

[Hàm kiểm tra số nguyên tố (hai số nguyên tố cùng nhau nếu UCLN bằng 1)](http://codepad.org/brGR7kkE)

[Hàm tìm ước chung lớn nhất sử dụng thuật toán Euclide](http://codepad.org/0K1OYpK6)

[Hàm tính nhanh an với n nguyên dương](http://codepad.org/bUfLxg7D)

[Hàm sinh số ngẫu nhiên (Random).](http://ideone.com/VzZOI9)

Chú ý: C++ 11 đã hỗ trợ kiểu dữ liệu random\_device trong thư viện random để phát sinh số ngẫu nhiên. Kiểu dữ liệu mt19937 có miền giá trị 32bit, mt19937\_64 có 64bit (miền giới hạn rộng hơn random cũ).

Vd:

Random\_device *tênbiến1*;

Mt19937 *tênbiến2*(*tênbiến1*());

uniform\_int\_distribution<int> tênbiến3(min, max);

auto *num*=tênbiến3(tênbiến2);

Cout<<num;

Nếu kết quả âm nghĩa là bị tràn số do kiểu int có miền giá trị nhỏ hơn. Thay kiểu in bằng auto thì biến num sẽ tự động chuyển kiểu cho phù hợp.

Để tạo số ngẫu nhiên trong khoảng giá trị min max nào đó, ta dùng: uniform\_int\_distribution<int> tênbiến3(min, max);

1. **ĐỆ QUY (Recursion):**

* Một **vấn đề mang tính đệ quy** nếu nó có thể được giải quyết thông qua kết quả của chính nó với đầu vào đơn giản hơn.
* **Hàm đệ quy***:* có lời ***gọi lại chính nó*** trong thân hàm *(có thể gọi thông qua một hàm con khác có chứa nó).*
* **Số lần gọi phải có giới hạn** (*điểm dừng*) => trường hợp cơ bản (base case): là một *input đủ nhỏ* để ta có thể giải quyết vấn đề mà không cần lời gọi đệ quy.
* **Phân loại**: 4 (*?*)
* ***Đệ quy tuyến tính*** **(***Linear Recursion***)**: *gọi lại chính nó 1 lần*, ngay cả khi lời gọi đệ quy xuất hiện nhiều lần thì cũng chỉ có 1 lần được chạy. Dễ chuyển sang vòng lặp => khử đệ quy. Vòng lặp chạy nhanh hơn.
* ***Đệ quy phi tuyến (****Multiple Recursion****)***: *gọi lại chính nó nhiều lần*. Trường hợp đặc biệt gọi lại 2 lần => ***đệ quy nhị phân*** (dãy Fibonacci). Khó chuyển sang cấu trúc lặp.
* ***Đệ quy hỗ tương*** (*Mutual Recursion* – Đệ quy gián tiếp – *Indirect Recursion*): *gọi lại chính nó thông qua 1 hoặc nhiều hàm khác chứa nó* => hiếm gặp.
* ***Đệ quy lồng*** (*Nested Recursion*): tham số trong lời gọi đệ quy là một lời gọi đệ quy. *Đệ quy lồng chiếm bộ nhớ rất nhanh*.

*Hoặc có thể chia thành* ***đệ quy đầu*** *và* ***đệ quy đuôi,*** *trong đó* ***đệ quy đuôi*** *có số lần chạy bằng ½ so với đệ quy thông thường.*

* **Call Stack và đệ quy:**
* Mỗi lời gọi đệ quy tạo ra 1 phần tử mới trong Stack
* C++ luôn *chạy phần tử ở đỉnh* của Stack trước
* Tại 1 thời điểm, Stack chỉ có thể chứa số lượng phần tử có hạn
* Khi chiều cao của Stack quá lớn, chương trình có thể gặp lỗi (*Stack Overflow*)

**Một số bài trong sách 1000 bài tập KTLT của thầy Khang (có đáp án tham khảo)**

[Bài toán điển hình về dãy Fibonacci: Tìm phần tử thứ n (giả sử 2 phần tử đầu tiên trong dãy là 1)](http://codepad.org/01hY9cDq)

[ĐỀ BÀI: Tính tổng 1 + 2 + 3 + ... + n](http://codepad.org/1RuGuV0D)

[ĐỀ BÀI: Tính tổng 1 + 1/2 + 1/3 + ... + 1/n](http://codepad.org/M44J9Dcp)

[ĐỀ BÀI: Tính 1/2 + 2/3 + … + n/(n+1))](http://codepad.org/Q1yZgpIT)

[ĐỀ BÀI: Tính 1/(1\*2) + 1/(2\*3) + … + 1/(n\*(n+1))](http://codepad.org/hyO5AgE2)

[ĐỀ BÀI: Tính 1 + 1\*2 + 1\*2\*3 + … + 1\*2\*3\*…\*n](http://codepad.org/W5TkBeMd)

[ĐỀ BÀI: Tính x^n](http://codepad.org/rDHlgI4k)

[ĐỀ BÀI: Tính tổng 1 + 2^2 + 3^2 + ... + n^2](http://codepad.org/7kCB7H7T)

[ĐỀ BÀI: Đếm số lượng chữ số của một số nguyên N](http://codepad.org/TDmH72ct)

[ĐỀ BÀI: Tính tổng các chữ số của số nguyên dương n](http://codepad.org/SeHS765B)

[ĐỀ BÀI: Tìm chữ số đầu tiên của số nguyên dương n](http://codepad.org/vlBDufXs)

[ĐỀ BÀI: Tìm số đảo ngược của số nguyên dương n](http://codepad.org/vlBDufXs)

[ĐỀ BÀI: Tìm chữ số lớn nhất/nhỏ nhất của số tự nhiên n](http://codepad.org/U2UKmSxz)

[ĐỀ BÀI: Cho biết dãy các chữ số của số nguyên dương n có toàn lẻ/chẳn hay không?](http://codepad.org/B1gIR4Fm)

[ĐỀ BÀI: Nhập/xuất mảng một chiều, tính tổng/tích các phần tử trong mảng, tìm phần tử lớn/nhỏ nhất, đếm số lượng số chẵn/lẻ, sắp xếp tăng/giảm các phần tử trong mảng.](http://codepad.org/COwohIFG)

[ĐỀ BÀI: Nhập/xuất mảng hai chiều, tính tổng/tích các phần tử trong mảng, tìm phần tử lớn/nhỏ nhất, đếm số lượng số chẵn/lẻ, sắp xếp tăng/giảm các phần tử trong mảng.](http://codepad.org/KC8SxgJh)

1. **MẢNG:**

* ***MẢNG 1 CHIỀU***: là một kiểu *dữ liệu có cấu trúc* do *người lập trình định nghĩa*; biểu diễn một *dãy các phần tử (biến) giống nhau (cùng kiểu)*; ***kích thước*** được ***xác định ngay khi khai báo*** và không bao giờ thay đổi; được C/C++ chỉ định một khối nhớ liên tục.
* Khai báo: *tường minh* hoặc *không tường minh* (dùng typedef).
* Có thể tự động xác định số lượng phần tử (vd int arr[] = {10, 2, 1998} => số lượng phần tử = 3)
* Khởi tạo giá trị 0 cho mảng vd: int **arr[4]= {0}** **⬄ arr[0] = arr[1] = arr[2]= arr[3] = 0**
* Khởi tạo giá trị cho một số phần tử đầu mảng vd **int arr[4] = {20, 55}** ⬄ **arr[0] =20, arr[1] = 55, arr[2]= 0, arr[3] = 0 => *tự động thêm 0*** *cho các phần tử còn lại.*
* Khởi tạo giá trị cho mọi phần tử của mảng vd int arr[3] = {10, 2, 1998}
* Truy xuất thông qua chỉ số mảng (index) vd int arr[3] = {10, 2, 1998}=> arr[0] = 10,…

***Một số lỗi thường gặp:***

* Khai báo ***không có số lượng phần tử*** trong mảng: vd ***int a[];*** *=>* ***int a[30];***
* ***Khởi tạo cách biệt với khai báo***: vd ***arr[3]; a = {10, 2, 1998};***

=> ***arr[3] = {10, 2, 1998};***

*Hoặc* ***arr[] = {10, 2, 1998};***

* ***Số lượng phần tử*** là biến hoặc hằng thường (const)

=> dùng chỉ thị tiền xử lí #define

* Chỉ số mảng không hợp lệ => chỉ số hợp lệ từ 0 -> n-1.
* Chỉ số mảng là ***số nguyên***.
* **Truyền mảng cho hàm:**

void **Function**(int a[30], int n);

void **Function** (int a[], int n);

void **Function** (int \*a, int n);

* Số lượng phần tử thực sự truyền qua biến khác (n)
* **Truyền mảng và lời gọi hàm**

void **ArrayInput**(int a[], int &n); => tham chiếu để lưu giá trị cho biến n

void **ArrayOutput**(int a[], int n);

void main()

{

int a[30], n; //phải *xác định số lượng phần tử ngay lúc khai báo*

**ArrayInput** (a, n);

**ArrayOutput** (a, n);

}

* **Lấy địa chỉ phần tử:** thêm & vd &a[0], &a[1],…
* ***MẢNG 2 CHIỀU:*** (ma trận) *tương tự mảng 1 chiều*
* **Khai báo**: ***(typedef) <tên kiểu cơ sở> <tên mảng> [số dòng][số cột]***

**Lưu ý**: *thứ tự số dòng, số cột là do mình quy định, có thể đổi lại nhưng sẽ gây khó khăn và dễ nhầm lẫn trong lúc lập trình vì trái quy ước thông thường.*

* 0 ≤ **chỉ số dòng**≤ ***số dòng*** - 1
* 0 ≤ **chỉ số cột** ≤ ***số cột*** - 1

Vd: khai báo int a[2][2] thì truy cập hợp lệ là a[0][0], a[0][1], a[1][0], a[1][1]

* **Số lượng phần tử tối đa** trong mảng là: **số dòng \* số cột**
* **Lấy địa chỉ**: thêm dấu **&** phía trước phần tử đó
* **Truyền mảng và lời gọi hàm*:***
* ***void* NhapMaTran *(int a[50][100], int &dong, int &cot);*** // đầy đủ
* ***void* NhapMaTran *(int a[][100], int &dong, int &cot);*** // bỏ ***số dòng*** tối đa *vì dựa vào kích thước cột mà trình biên dịch sẽ suy ra kích thước dòng.*
* ***void* NhapMaTran *(int (\*a)[100], int &dong, int &cot);*** //con trỏ
* **Gọi hàm**: **NhapMaTran(a, dong, cot);**

**Lưu ý:** C/C++ ***không hỗ trợ phép gán “=” cho hai mảng***

Vd: int a[10][2], b[10][2];

a = b; => **sai**

*sửa*: *for(int i=0; i < 10; i++)*

*for(int j=0; j < 2; j++)*

*a[i][j] = b[i][j];*

* **Khởi gán mảng 2 chiều:**

+vd: **int MaTran[3][2]={ {1, 2}, {3, 4}, {5, 6} }**

⬄ ***MaTran[0][0] = 1, matran[0][1] = 2, matran[1][0] = 3, matran[1][1] = 4, matran[2][0] = 5, matran[2][1] = 6***

+Có thể bỏ cặp dấu **{}** của các cặp phần tử int **MaTran[3][2]={ 1, 2, 3, 4, 5, 6}**

+ *Nếu không đủ số lượng phần tử tối đa thì trình biên dịch sẽ tự động thêm số* ***0*** *và các vị trí còn thiếu.*

1. **CHUỖI KÝ TỰ:**

***a) Trong C:***

* Kiểu char chứa **1 ký tự** => dùng *mảng các phần tử kiểu char* để lưu chuỗi ký tự.
* Ký tự **‘\0’**: kết thúc chuỗi (*mã ASCII là 0*)
* Mảng **n** phần tử lưu *tối đa* **n-1** ký tự, đánh **chỉ số** từ **0** tới **n-2** *do phần tử cuối mảng phải là* ‘**\0’ (***có chỉ số là n-1***)**
* vd: ***char chuoi[10]= “tam”*** thì *chuoi[0]= ‘t’*, *chuoi[1]= ‘a’*, *chuoi[2]= ‘m’*, *chuoi[3]= ‘\0’*. Nếu gán ***chuoi[0] = 0*** hay ***chuoi[0] = ‘\0’*** thì chuỗi ***chuoi*** xem như ***rỗng***, những ký tự có trong chuỗi xem là ***rác***
* Để in ký tự **“\”** trong chuỗi hằng thì viết **2 lần** ký tự này
* Phải **chủ động** kiểm soát số lượng ký tự tối đa trong chuỗi
* **Không thể***: gán giá trị, sử dụng phép +, >, <, => sử dụng hàm trong thư viện* **<string.h>** *hoặc tự viết hàm.*

**Một số hàm thông dụng:**

* **strlen()** : hàm tính độ dài chuỗi ký tự
* **strcpy()**: hàm sao chép chuỗi ký tự
* **strdup()**: hàm tạo bản sao
* **strlwr()/strupr()**: hàm chuyển chuỗi thành chuỗi viết thường / hoa
* **strrev()**: hàm đảo ngược
* **strcmp()**: hàm so sánh 2 chuỗi có phân biệt hoa thường
* **stricmp()**: hàm so sánh 2 chuỗi không phân biệt hoa thường
* **strcat()**: hàm nối 2 chuỗi
* **strchr()**: tìm xem ký tự ***c*** có trong chuỗi hay không
* **strstr()**: hàm tìm chuỗi trong chuỗi
* **strtok()**: tách từ trong chuỗi ***s*** dựa vào một trong các ký tự ngắt

***Xóa bộ nhớ đệm****:* ***fflush(stdin);***

***Lưu ý:***

* Nhập chuỗi:
* Dùng **scanf()** với đặt tả là **%s** => nhận các ký tự đến khi *gặp khoảng trắng thì dừng*
* Dùng hàm **gets()** nhận các ký tự đến khi *gặp ký tự xuống dòng, nhận luôn khoảng trắng*
* Xuất chuỗi:

- Dùng hàm **printf()** với đặt tả là **%s** *=> không xuống dòng*

**-** Dùng hàm **puts()** =>tự động xuống dòng

b)Trong C++: được hỗ trợ kiểu **string** trong thư viện chuẩn STL

* **#include <string>**
* **Khai báo**: **string s1;**

**string s2(“vo minh tam”);**

**string s3 = “vo minh tam”;**

* **Nhập xuất**: dùng **cin** (gặp khoảng trống thì dừng)**, cout**
* *Để nhập một dòng gồm cả khoảng trắng dùng* **getline(cin,** *<tên chuỗi>***);**
* *Để nhập nhiều dòng =>* ***getlline(cin, <tên chuỗi>, “-”)***: kết thúc nhập khi gặp ký tự “-”.
* *Xóa bộ nhớ đệm:* ***cin.ignore(32767, “\n”);*** *hoặc* ***cin.ignore();***
* Phép **+** để ghép chuỗi
* Phép gán **=** để gán chuỗi a cho chuỗi b
* Các phép so sánh: **==, !=, >, <, >=, <=**
* *Phương thức* **length()** *(vd s.length()) lấy kích thước chuỗi* và phép lấy chỉ số**[]**

[ĐỀ BÀI: Xác định năm âm lịch bằng phương pháp dùng bảng tra cho C](http://ideone.com/QZFeor)

[ĐỀ BÀI: Xác định năm âm lịch bằng phương pháp dùng bảng tra cho C++](http://ideone.com/hDvqFV)

[ĐỀ BÀI: Chuẩn hóa chuỗi (tên)](https://ideone.com/LGd15B)

1. **DỮ LIỆU CẤU TRÚC:**

*Gom chung các thông tin của cùng một đối tượng để dễ dàng khai báo, truy xuất và xử lí.*

***Khai báo tường minh:***

**struct** <tên kiểu cấu trúc>

**{**

<kiểu dữ liệu> <tên thành phần 1>

…

<kiểu dữ liệu> <tên thành phần n>

**}**<tên biến 1>, <tên biến 2>**;**

***Khai báo không tường minh:***

**struct** <tên kiểu cấu trúc>

**{**

<kiểu dữ liệu> <tên thành phần 1>

…

<kiểu dữ liệu> <tên thành phần n>

**};**

**struct** <tên kiểu cấu trúc> <tên biến>**;**  *//C++ có thể bỏ từ khóa struct*

***Ngoài ra có thể dùng từ khóa typedef để định nghĩa kiểu dữ liệu mới:***

**typedef struct** <tên kiểu cấu trúc>

**{**

<kiểu dữ liệu> <tên thành phần 1>

…

<kiểu dữ liệu> <tên thành phần n>

**};**

<tên kiểu cấu trúc> <tên biến>**;**

***(Khi thao tác với CÂY nên dùng từ khóa typedef để định nghĩa cây)***

***Có thể gán giá trị cho các thành phần của biến trong struct ngay lúc khai báo***

struct <tên kiểu cấu trúc>

{

<kiểu dữ liệu> <tên thành phần 1>;

…

<kiểu dữ liệu> <tên thành phần n>;

} <tên biến> = {<giá trị 1>, …, <giá trị n>};

***Hoặc***

typedef struct

{

<kiểu dữ liệu> <tên thành phần 1>;

…

<kiểu dữ liệu> <tên thành phần n>;

}<tên kiểu cấu trúc>;

<tên kiểu cấu trúc> <tên biến> = {<giá trị 1>, … ,<giá trị n>};

***Không thể truy xuất trực tiếp, phải thông qua toán tử chấm “.” (dot operation)***

***Gán giá trị cho biến cấu trúc:***

*<****biến đích****> = <****biến nguồn****>*

*<****biến đích****>***.***<****tên thành phần****> = <****giá trị****>*

* ***Thành phần của cấu trúc có thể là một cấu trúc khác (lồng nhau)***
* ***Thành phần của cấu trúc có thể là mảng***
* ***Thành phần của cấu trúc có kích thước theo bit***
* ***Cấu trúc đệ quy tự trỏ***

*Đặc biệt: C++ cho phép định nghĩa các phép toán cơ sở cho kiểu dữ liệu mới (****operator overloading****) thay vì viết các hàm như trước đây, mục đích là để dễ hiểu, dễ thao tác vì nó gần giống với ngôn ngữ tự nhiên của con người.*

***Cách thực hiện:***

*<****kiểu trả về****> operator* **X***(các tham số)*

***{***

*… // Xử lý bình thường*

***}***

*Với* **X** *là các toán tử có thể nạp chồng được: +, -, \*, /, +=, -=, =, …*

1. **XỬ LÝ VỚI TẬP TIN VĂN BẢN:**

Có ***2 loại tập tin chính***: ***tập tin văn bản thô và tập tin nhị phân***

* **Tập tin văn bản thô**
* Thường lưu dưới dạng **\*.txt**
* Gồm nhiều dòng, ngăn cách bởi ký hiệu ngăn cách dòng **(\r, \n)**
* Quy ước kêt thúc bằng ký hiệu kết thúc tập tin (*end of file – EOF character*)
* ***Tập tin văn bản thô mở rộng***: có thể lưu ký tự Unicode và ký tự nhiều byte
* **Tập tin nhị phân**

Là những tập tin không có cấu trúc như tập tin văn bản thô.

* **Tên tập tin và đường dẫn thư mục:**
* Mỗi tập tin có một tên duy nhất tuân theo quy tắc đặt tên của hệ điều hành:

***<tên tập tin>.<phần mở rộng>***

* **Đường dẫn**: theo quy ước của C/C++ thì dấu **‘\’** là ký tự đặc biệt phải được viết 2 lần, vd **“C:\\Data\\file.txt”,** còn nếu *người dùng cuối nhập vào lúc chạy chương trình* thì vẫn *ghi bình thường* **“C:\Data\file.txt”.** *Nếu phần mở rộng không ghi gì hết “C:\Data\file” thì file của bạn sẽ thuộc loại File Folder (vào Properties xem Type of File)*

*Thông thường file của ta nên được đặt cùng chỗ với chương trình, vì khi đem chương trình qua các môi trường khác nó vẫn chạy được! vd* ***“file.txt”.***

*Để tạo file trong cùng nơi chứa source code: nhấp phải project -> Open Folder in File Explorer -> tạo file theo mong muốn.*

* **LÀM VIỆC VỚI C:**
* Kiểu dữ liệu: **FILE\*** trong **<stdio.h>**
* Hàm:
* **fopen(<***tên tập tin***>, <***đối số***>)**: mở tập tin

vd: ***FILE\* f = fopen(“data.txt”, “rt”);***

**Với các đối số**:

* **“r”** hay **“rt”** mở file để đọc, nếu không có trả về NULL;
* **“r+”** hay **“r+t”** mở để đọc và ghi, được tạo mới nếu chưa có, dữ liệu đã có không bị xóa;
* **“w”** hay **“wt”** mở để ghi (từ cuối file), tạo mới nếu chưa có, nếu có rồi thì toàn bộ dữ liệu bị xóa;
* **“a”** hay **“at”** mở để ghi dữ liệu vào cuối file, tạo mới nếu chưa có;
* **“w+”** hay **“w+t”** giống **“w”** nhưng bổ sung thêm tính năng đọc;
* **“a+”** hay **“a+t”** giống **“a”** nhưng bổ sung thêm tính năng đọc (từ đầu file).

***Ngoài ra còn có các chế độ xử lí tập tin nhị phân***: **“rb”, “rb+”, “wb”, “wb+”, “ab”, “ab+”, “r+b”, “a+b”, …**

*Một cách dễ nhớ:* **r** *viết tắt của read (đọc),* **w** *viết tắt của write (viết),*

**t** *viết tắt của text (tập tin văn bản),* **a** *viết tắt của append (bổ sung thêm),*

**b** *viết tắt của binary (nhị phân).*

* **fclose(<***tên tập tin***>);** cập nhật dữ liệu và đóng tập tin vd: ***fclose(f);***
* **fcloseall()/ \_fcloseall();** đóng tất cả tập tin hiện hành
* **fgets(***<chuỗi>***,** *<số lượng n ký tự>***,** *<tên tập tin>***);** đọc ***n-1*** ký tự từ ***chuỗi*** vào tập tin (**đọc từng dòng**)
* **fgetc(***<tên tập tin>***);** lấy một ký tự (unsigned char) trong tập tin và tăng vị trí của cn trỏ trong tập tin lên 1 đơn vị (**đọc từng ký tự**)
* **fscanf(***<tên tập tin>, <định dạng>, <tên biến>,…***);** đọc dữ liệu có *định dạng* từ *tập tin* vào *tên biến, với đặt tả* ***%s*** *thì chỉ đọc từng từ của dòng hiện hành mà bỏ qua các khoảng trống*
* **fwrite(***<tên biến>, <tổng kích thước>, <kích thước 1 phần tử >, <tên tập tin>***);** ghi dữ liệu từ *tên biến* vào *tập tin*
* **fread(***<tên biến>, <kích thước 1 phần tử>, <tổng kích thước>, <tên tập tin>***);** đọc dữ liệu từ *tập tin* vào *tên biến*
* **fputc(***<ký tự>, <tên tập tin>***);** ghi *ký tự* vào *tập tin*
* **fputs(***<chuỗi>, <tên tập tin>***);** ghi *chuỗi* vào *tập tin*
* **fprintf(***<tên tập tin>, <định dạng>,<tên biến>,…* **);** đọc dữ liệu từ *tên biến* có *định dạng* vào *tập tin.* Nếu *tên tập tin* là **stdout** thì hàm này giống như **printf()**
* **feof(***<tên tập tin>***);** trả về 1 nếu đến cuối tập tin, ngược lại trả về 0

*Về cơ bản, khi chương trình kết thúc thì các tập tin đang mở sẽ tự động được đóng, nhưng ta nên đóng một cách tường minh để tránh các sự cố ngoài ý muốn.*

***Khi gọi các hàm thao tác trên tập tin thì một số công việc được thực hiện trên vùng đệm dữ liệu (vùng nhớ tạm). Ta có thể cập nhật dữ liệu vào tập tin mà không cần đóng tập tin bằng hai hàm:***

* Cập nhật lên *tập tin đang mở*: **fflush(<***tên tập tin***>);**
* Cập nhật lên *tất cả tập tin đang mở*: **flushall();**
* Truy xuất: tuần tự (sequentical access) và ngẫu nhiên (random access)
* **rewind(**tên tập tin**);** dời vị trí đọc về đầu
* **fseek(***fp, offset, origin***);** dời vị trí con trỏ chỉ vị trong *stream fp* với vị trí *offset* *(thường là 0 - độ dịch tính theo byte)* so với cột mốc *origin*

*origin*: *SEEK\_SET* (hay 0) đầu tập tin

*SEEK\_CUR* (hay 1) vị trí hiện tại

*SEEK\_END* (hay 2) cuối tập tin

* **ftell(***tên tập tin***);** : trả vị trí đọc hiện tại

vd: ***int filesize = ftell(fp);*** *// trả về số byte tính từ đầu file đến con trỏ file hiện tại*

* Các thiết bị nhập xuất chuẩn định nghĩa sẵn:
* **stdin**: nhập chuẩn, mặc định là bàn phím
* **stdout**: xuất chuẩn, mặc định là màn hình
* **stderr**: lỗ chuẩn, mặc định là màn hình
* **stdprn** (MS-DOS): in chuẩn (máy LPT1:)
* **stdaux** (MS-DOS): phụ chuẩn, cổng COM 1:

**Lưu ý nhẹ**: Nếu không dùng được một trong các hàm trên (*code trên C không chuẩn*) thì khi biên dịch chương trình, trình biên dịch sẽ thông báo và gợi ý hàm. Ví dụ *scanf -> scanf\_s; fscanf -> fsanf\_s; getch() -> \_getch(); fopen(…) -> fopen\_s(…); …*

Đặc biệt, với hàm **fopen\_s(…)** thì cách dùng có khác đôi chút, , các bạn vào link sau để xem hướng dẫn <https://msdn.microsoft.com/en-us/library/z5hh6ee9.aspx>

Đơn giản là: dùng từ khóa **errno\_t** *<tên biến>*;

*<tên biến>* = **fopen\_s**(&tên tập tin, “tên file”, “chế độ”);

**Kiểm tra xem file đã được mở chưa***:* ***if(****tên biến ==0****)*** *=> đã được mở (tức là không có lỗi);*

Vào link sau để xem bài mẫu (mình lấy từ trang daynhauhoc.com – cho em xin phép luôn ạ) <https://daynhauhoc.com/t/cach-su-dung-ham-fopen-s/7293/7>

*Một số đuôi của tập tin: .txt (notepad), .dat (dữ liệu thô – tìm hiểu thêm), .doc/ .docx (word), .xls (excel), .png/ .jpg/ .gif (ảnh), .zip/ .rar (nén), .mp3 (âm thanh), …*

*Nếu chia hàm để xử lý thì khi đọc dữ liệu từ tập tin vào biến bằng hàm hoặc mở tập tin bằng hàm thì phải truyền tham chiếu cho biến và tập tin đó (&).*

* **LÀM VIỆC VỚI C++:**
* Dùng thư viện: **<fstream>**

**cin >>** ----------------- **ifstream >>**

**cout <<** ----------------- **ofstream <<**

*riêng* ***fstream*** *có thêm sức mạnh của cả* ***ifstream*** *và* ***ofstream***

* **Tạo** file và **đọc** dữ liệu ***từ tập tin***: *(dùng lớp ifstream)*

**ifstream** <*tên biến file*> **(“**<*đường dẫn*>**”);**

*hoặc* **ifstream** <*tên biến file*> **(“**<*đường dẫn*>**”,** *<chế độ>***);**

*hoặc* **ifstream** *<tên biến file>***;**

*<tên biến file>***.open(“**<*đường dẫn*>**”,** *<chế độ>***);**

***<tên biến file>*****>>*****(dữ liệu)*;**

* **Tạo** file và **ghi** dữ liệu ***vào tập tin***: *(dùng lớp ofstream)*

**ofstream** <*tên biến file* > **(**<*đường dẫn*>**);**

*hoặc* **ofstream** <*tên biến file*> **(“**<*đường dẫn*>**”,** *<chế độ>***);**

*hoặc* **ofstream** *<tên biến file>***;**

*<tên biến file>***.open(“**<*đường dẫn*>**”,** *<chế độ>***);**

***<tên biến file>*****<<*****(dữ liệu)*;**

* ***Với các chế độ (mode)***: nguồn http://vietjack.com/cplusplus/file\_io\_trong\_cplusplus.jsp

|  |  |
| --- | --- |
| Chế độ | Miêu tả |
| ios::app | Chế độ Append. Tất cả output tới file đó được phụ thêm vào cuối file đó |
| ios::ate | Mở một file cho output và di chuyển điều khiển read/write tới cuối của file |
| ios::in | Mở một file để đọc |
| ios::out | Mở một file để ghi |
| ios::trunc | Nếu file này đã tồn tại, nội dung của nó sẽ được cắt (truncate) trước khi mở file |

***Chúng ta có thể linh động trong việc kết hợp các chế độ với nhau phù hợp với mục đích lập trình bằng cách hoặc (* | *) các chế độ lại với nhau.***

***Ví dụ***: mở file để đọc và ghi như sau

***fstream file;***

***file.open(“filename.dat”, ios::in | ios::out);***

* ***Kiểm tra tập tin có mở được hay không:***

***if (! <tên biến******file >)*** *hoặc* ***if (<tên biến******file > == NULL)***

*hoặc dùng hàm có sẵn* ***if(<tên biến file>.fail() ==true)***

***cout<< “Không mở được file!”;***

* ***Đóng tập tin:*** <*tên biến file* >**.close();**
* ***Kiểm tra cuối tập tin*** dùng hàm Boolean **eof()**: <*tên biến file* >**.eof();**
* ***Đọc tập tin trên từng dòng*** dùng hàm **getline(…);**

Vd: *lấy từng dòng trong file và xuất ra màn hình*

***string str;***

***ifstream filein("check.txt");***

***if(!filein)***

***{***

***cout << "\nMở file lỗi!";***

***return 1;***

***}***

***for(str; getline(filein, str); ) //****lấy từng dòng trong* ***filein*** *đưa vào biến* ***str***

***cout << endl << str;***

***filein.close();***

* ***Con trỏ vị trí trong file:***

*Tương tự như C, các lớp đối tượng ifstream và ofstream trong C++ đều cung cấp các hàm thành phần cho việc xác định vị trí của con trỏ trong file. Con trỏ vị trí file là một giá trị integer tính từ đầu file đó*

* Trong ifstream: seekg(<tham số 1>, <tham số 2>);
* Trong ofstream: seekp(<tham số 1>, <tham số 2>);

Với: <tham số 1>: thường là 1 giá trị long int

<tham số 2>: chỉ dẫn hướng tìm kiếm, có thể là

* **ios::beg** (*mặc định*): liên quan tới vị trí bắt đầu
* **ios::cur**: liên quan tới vị trí hiện tại
* **ios::end**: liên quan tới phần kết thúc

*Như đã nói, fstream có thể khai báo, mở file để đọc và ghi mà không cần phân biệt như ifstream và ofstream, cách sử dụng hoàn toàn tương tự.*

**Các bạn xem một số ví dụ sau để hiểu rõ hơn:**

* + - 1. [Đọc dữ liệu từ tập tin và xuất ra màn hình.](http://ideone.com/iLQPn6)
      2. [Ghi dữ liệu vào tập tin.](http://ideone.com/QHZwqD)
      3. [Tính tổng số tiền điện, tiền nước mà chủ nhà A20 trong ký túc xá ĐHQG HCM thu được trong tháng 7/2017, biết dòng đầu tiên trong file Tiendiennuoc\_T7.txt là tổng số phòng, mỗi dòng tiếp theo gồm số tiền điện và số tiền nước được cách nhau bằng khoảng trắng. Xuất vào file QuanliA20\_T7.txt.](http://ideone.com/48j1n3)

1. **MỘT SỐ VẤN ĐỀ VỀ CON TRỎ VÀ KĨ THUẬT BỘ NHỚ ĐỘNG:**
2. **MẢNG ĐỘNG TRONG THƯ VIỆN CHUẨN (S***tandard* **T***emplate* **L***ibrary* của **C++ )**

* Thư viện hỗ trợ: **#include <vector>**
* Ưu điểm: ***kích thước mảng tự động cập nhật***
* Cú pháp: **vector <T>** *<tên biến>***;** *với* **T** *là một kiểu dữ liệu cụ thể*

*Vd: vector <int> a;*

* *Lưu ý: toán tử* **&** *(đặt trước tên biến)*
* *Các phương thức hỗ trợ:* ***<tên biến>*.*<tên phương thức>***

|  |  |
| --- | --- |
| Phương thức | Ý nghĩa |
| size() | Trả số phần tử hiện có trong mảng. |
| resize(*int newsize*) | Thay đổi kích thước mảng, quy định bởi tham số *newsize*; nếu kích thước mới nhỏ hơn kích thước cũ -> mảng co lại, một số phần tử bị xóa. |
| push\_back(T *key*) | Thêm phần tử key có kiểu T vào cuối mảng, kích thước mảng tự động tăng thêm 1 đơn vị. |
| pop\_back() | Xóa phần tử cuối cùng của mảng, kích thước mảng tự động giảm đi 1 đơn vị. |